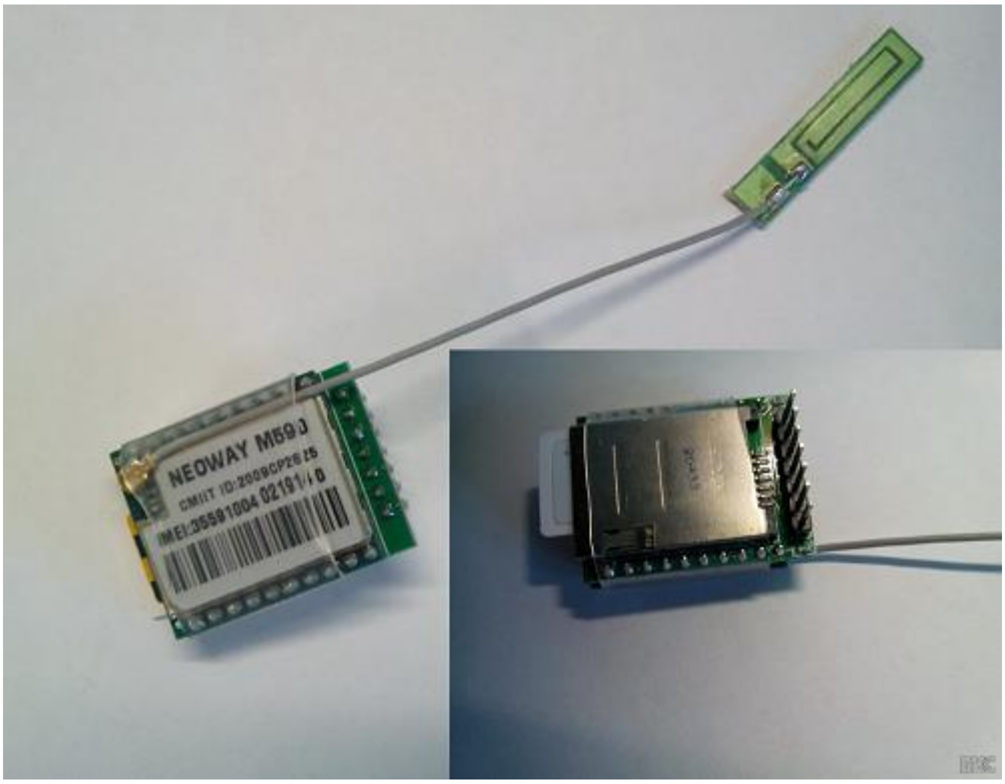
USING THE NEOWAY M590E GPRS MODEM

The Neoway M590E GPRS modem provides mainly SMS services and GPRS connectivity. It supports TCP (server or client mode), UDP, FTP and DNS checks. Like most modems, it uses AT commands for interaction with a host, though some commands are specific to this modem only. This documentation focuses mainly on using the module with an Arduino Uno as host. All documents relevant to using this modem can be found here:

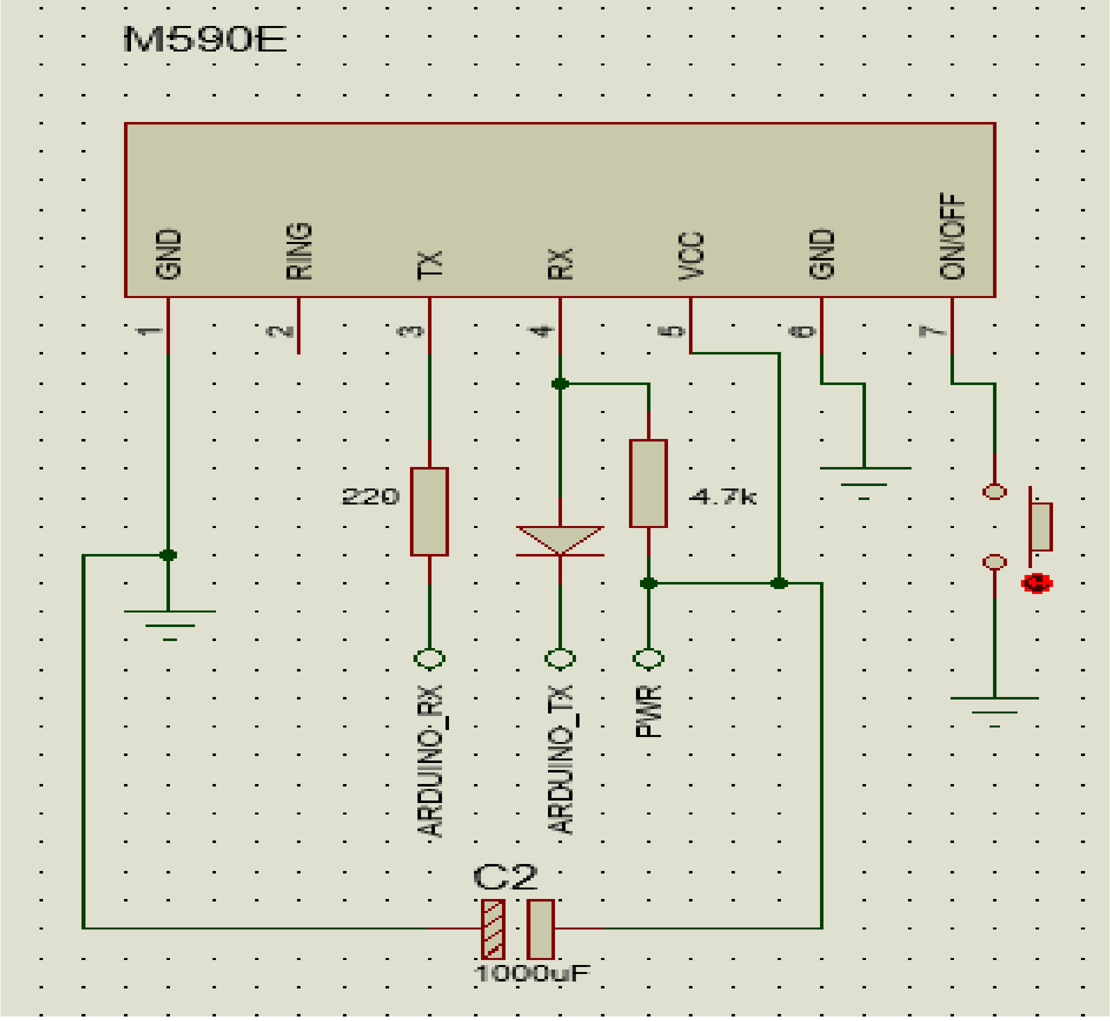
http://docs.mirifica.eu/Neoway.com/archive/M590/



REQUIREMENTS

- A steady 3.3 – 4.5 V power supply (ideally 3.9 V) that can supply at least 2A to the M590 module without dipping below 3.3 V.
- A 1000uF (at least) capacitor that can help supply the large instantaneous current. This should be placed close to the module. A 100uF tantalum capacitor will suffice, if you’re using a lithium battery as power supply.
- IN4001 diode, 200 Ω resistor, 4.7 KΩ resistor, pushbutton and a SIM card, of course, for SIM-related operations.

CONNECTIONS



In the diagram above,

ARDUINO_RX = RX pin of SoftwareSerial (e.g. Pin 2)

ARDUINO_TX = TX pin of SoftwareSerial (e.g. Pin 3)

PWR = Power supply (ideally, 3.9 V) for the GPRS modem

Note the ground symbols above indicate the common ground connection of the modem’s ground pin, the ground of its power supply, and the ground pin of the Arduino. You must connect all three GNDs together.

Pin 1 (GND) is the leftmost pin in the diagram above. You can verify this by testing for continuity between Pin 1 and Pin 6 (next-to-last pin on the right) with your multi-meter. The 2 pins are already internally connected as GND pins.

POWER-ON PROCEDURE

For this project, a 3.7 V LiPo battery was used to power the module. The 1000uF capacitor was placed as close to the module as possible, between V_{CC} and GND. The Arduino (or whatever MCU you're using) must first be powered up before supplying power to the module. After both devices have been supplied with power, you can then turn on the module.

Here, the function of the ON/OFF pin will be explained. The ON/OFF pin is used to turn on or turn off the GPRS modem. It is internally pulled HIGH to slightly above 2 V. It is an active LOW pin. This means that to turn on the module, the pin must be sent LOW. The duration of this low signal should be at least 500ms, after which the pin can be pulled back HIGH. If another LOW signal (of 500ms) is sent, the module turns off this time. Thus, successive pulses toggle the module between the on and off state. In the diagram above, the ON/OFF pin is connected to GND through a switch. Thus, every successive press of the switch (for 500ms at least) toggles the modem's state between ON and OFF. Therefore to turn on the module now, press the button for about a second and release.

The red LED should begin blinking once every second. This is an indication that the module is ON. The LED goes off if the module enters the SLEEP mode or is turned OFF.

POWER-OFF PROCEDURE

The module can be powered off using the ON/OFF pin, as explained previously. Simply press the button once more for 1 second. Wait 5 seconds at least for the module to log off the network and properly shut down and then you can disconnect it from its power supply.

In place of using the ON/OFF pin, a command can also be sent to the module to power it down. The command "AT+CPWROFF" is first sent to the module. Wait for 5 seconds as before and then disconnect the module from power.

The power-on and power-off procedures MUST be followed exactly to prevent incorrect operation or start-up failure of the module, or even corruption of the module's firmware. Make sure to use a very stable power supply (like a LiPo battery), one that is unlikely to spike unpredictably. Spikes can cause over-voltage conditions beyond the module’s capacity to handle, resulting in a short-circuit between the module’s V_{CC} and GND. When using a bench regulated power supply, make sure to disconnect the module first before turning the supply off.

SOFTWARE

Most GSM/GPRS modems are controlled using AT commands, though some AT commands are specific to the device manufacturer. ‘AT’ stands for ‘Attention” and all commands to this modem must include it as a prefix. Every AT command must end with the carriage-return character ‘\r’ to indicate the end of the command. The AT command set for the M590E can be found among the files at the link given earlier.

AT commands are sent to the modem via the UART protocol. However, when using the Arduino Uno, we need the hardware serial port for sending received packets to the PC (the Arduino IDE serial monitor, specifically) and for debugging. Thus, a serial port is created using software (a.k.a. SoftwareSerial), that will be used to transmit commands and receive responses from the modem.

Now that the medium of communication has been decided, we now have to choose the baud rate for SoftwareSerial. The M590E supports specific baud rates from 2400 to 460800 baud (consult AT command set). By default, an M590E modem is set to ‘auto-bauding’. This means that it will automatically detect the baud rate at which you send commands to it and adjust to that rate. This will only happen if the first command you send to it is ‘*AT*’. The response to this command (and most other commands), if successful, is ‘*OK*’. This indicates that the modem has adjusted to your current SoftwareSerial baud rate and will keep communicating with you at that same speed until you power it down. Auto-bauding is only supported for certain baud rates. Ensure to use multiples of 9600 baud, to be on the safe side. On first use, in your program, set the SoftwareSerial port baud rate to 115200, in case auto-bauding is not the modem’s default setting. You may choose to leave the modem setting to ‘auto-baud’ or you may set its baud rate to a specific baud rate permanently using commands. The command ‘AT+IPR?’ is a query that informs you if the modem is set to auto-baud or if it’s set to a particular baud rate. The response ‘MODEM: STARTUP’ is usually received immediately the modem is turned on. This message will be not be received if the modem is set to auto-bauding.

Once the modem has adjusted to the baud rate (verified by sending ‘AT’ and receiving ‘OK’), it begins authenticating the SIM card (if one is present). Once it’s done, you may receive a response of ‘+PBREADY’. Then the modem tries to register the SIM on the network. Only after registration can you use SIM-related commands like sending SMS and using GPRS. The command ‘AT+CREG?’ is used to determine the status of SIM registration. The response is usually in the format ‘+CREG x, y’. If y is 1 or 5, then the SIM has successfully connected to the network. Else, if y is 3 or anything else, then registration failed. Keep querying the modem with the command for about a minute, until you get the right response. If it doesn’t get registered within a minute, then try another SIM card (maybe from another network) or move to somewhere with better reception. The command ‘AT+CSQ’ can be used to determine the signal quality. Consult the AT command set on how to interpret the possible responses.

After the SIM has been registered, you can send SMS, make calls (without any audio, sorry), use GPRS functions, etc. (provided you have sufficient airtime, of course). If a SIM is not inserted, however, you can begin issuing commands as soon as you receive ‘OK’ from the first ‘AT’ sent to the module. SIM-related commands will obviously not work; they will return errors or yield no results.

To power down the modem, send the command ‘AT+CPWROFF’. You should receive ‘OK’ from the modem. Wait for 5 seconds before disconnecting the modem.

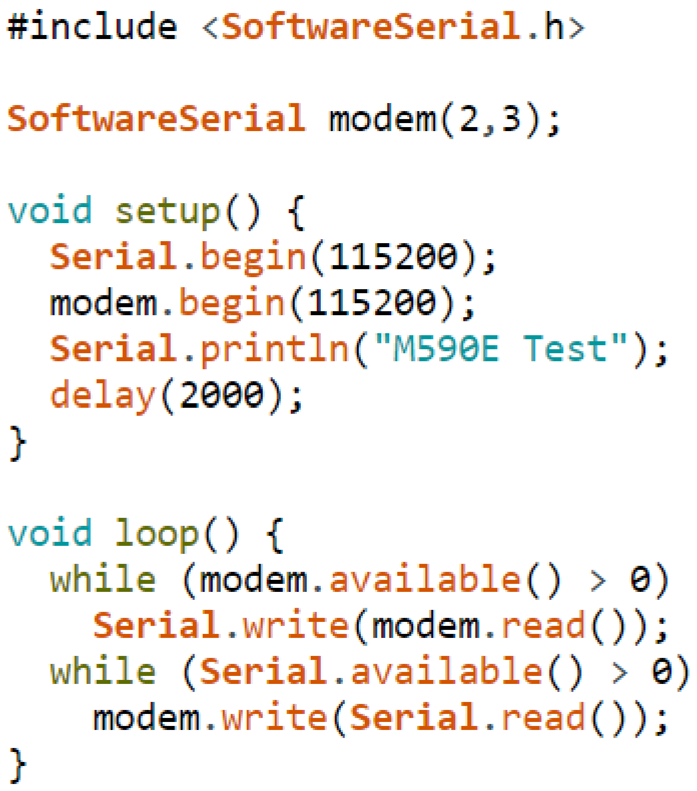
In order to accomplish all the ‘sending’ and ‘receiving’ of commands from the modem, a program must be uploaded to the Arduino to perform this task. The program works like this basically:

- You send an AT command from your PC to the Arduino using the hardware serial port; it’s easiest to do this by typing the command in the Arduino IDE serial monitor and pressing Enter. Make sure the serial monitor is always set to ‘Both NL & CR’ at the bottom, else your command will fail. The baud rate used by the hardware serial should be at least equal to that of the SoftwareSerial. Set the serial monitor to that baud rate.
- The Arduino forwards the commands to the modem unchanged through the SoftwareSerial port. In this way, the Arduino serves as a middle-man between the PC and the modem.
- The modem receives the commands, processes it and response; the response is received by the Arduino through the SoftwareSerial port.

- The Arduino forwards the response unchanged to the PC through the hardware serial port and the response is displayed on the serial monitor.

This cycle is followed for every command sent and every response received. The program simply transmits everything it receives from the PC to the modem, and transmits everything it receives from the modem to the PC.

This is the code:



```
#include <SoftwareSerial.h>

SoftwareSerial modem(2,3);

void setup() {
  Serial.begin(115200);
  modem.begin(115200);
  Serial.println("M590E Test");
  delay(2000);
}

void loop() {
  while (modem.available() > 0)
    Serial.write(modem.read());
  while (Serial.available() > 0)
    modem.write(Serial.read());
}
```

The SoftwareSerial library is included. A SoftwareSerial object named ‘modem’ is created using Arduino pins 2 and 3 as RX and TX respectively. The setup() function initializes the hardware and software serial ports to 115200 baud. This is the default baud rate, in cases where auto-bauding is not the factory default setting. SoftwareSerial is known to be unreliable at high baud rates like 115200 so it is recommended you reduce the baud rate to a safe value like 19200. The programs waits for 2 seconds to allow the modem to stabilize.

The loop() function implements the main function of the Arduino in this assembly: a communication channel between the PC serial monitor and the modem. Whenever there’s data available in the SoftwareSerial buffer, the data is written to the serial monitor. Whenever there’s data available in the hardware serial buffer (i.e. data that was typed in the serial monitor), the data is sent to the modem. This process continues until the Arduino is powered down.

With this program uploaded to the Arduino, your modem connected as indicated in the circuit diagram, communication established between the modem and PC via AT commands and SIM authentication and registration completed, you are now set to use the modem for any of its supported operations. Some sample operations will be explained below. Note that all AT commands must have a '\r' (carriage return) at the end, which should be attached automatically by your serial monitor whenever you send a command. In the examples below, the '\r' will be omitted for clarity and the commands are to be sent one after the other to the modem, after receiving the right response to each command. Commands are in italics, responses are in normal font style but indented. Comments have '//' before them.

- GET MODEM AND SIM INFO

```
AT+CGSN             //Show modem IMEI number

        358511020024166
        OK

AT+CCID             //Show SIM ICCID

        +CCID: 89860002190810001367
        OK

ATI                 //Show modem info

        NEOWAY
        M590
        REVISION 01.30e
        OK
```

Consult AT command set for more details.

By default, 'command echo' is enabled. This means when you send a command, the response is usually the command you sent plus the response. E.g.

```
AT

        AT
        OK
```

The command ‘AT’ was returned together with the response. To disable echo, if you want, send the command ‘ATE0’. The response is ‘OK’ as is the case for all other successful commands.

- **SEND SMS**

You need to enter this in a script and upload to the Arduino for reasons I will explain.

To send SMS, you must first set the SMS input mode to ‘text’ and set the character set to ‘GSM’. This must be done before reading or sending SMS or sending USSD codes. Enter the following commands:

```
AT+CMGF=1               //Set modem to text mode
        OK
AT+CSCS=”GSM”           //Set modem character set to ‘GSM’
        OK
```

You can find out if you have sufficient airtime by sending this command with the appropriate USSD code for checking account balance. E.g. for MTN SIMs:

```
AT+CUSD=1,”*556#”,15

        Your account balance is ……….
        OK
```

Just change the code in quotes above to the proper one for your network. You can use this command for any other USSD code supported by your network. Only send this command after sending the two commands preceding it.

Next, you send this command whose only argument is the recipient’s phone number with the country code preceding it.

```
AT+CMGS=”+2348034445555”

        >
```

After you get the ‘>’ character, you can now begin entering the message. The length of the SMS must not be greater than 140. At the end of the message, you must enter the key combination <Ctrl + Z>. It is equivalent to ASCII code 26. Unfortunately, there’s no way to enter this from the Serial monitor. The serial monitor converts everything you type to a string before sending so typing 26 would mean sending “26” and not the number 26, like we want. So you must enter all this code into a sketch and upload to the Arduino to run and display appropriate confirmation at each stage. If your SMS sends successfully, you’ll receive a reply like this:

```
        +CMGW: 15
        OK
```

Else, you’ll receive an error like “+CMS: ERROR”. Ensure the SIM is registered on the network, there’s sufficient airtime and you’ve followed all the steps exactly.

CONNECT TO A WEBSITE/SERVER

The M590E has GPRS connectivity that can be used to connect to the internet through the network provider and access websites or even server programs you’ve set up on your PC. Begin by sending the following commands:

```
AT+XISP=0                     //Select the internal protocol stack of the modem
        OK
```

```
AT+CGDCONT=1,”IP”,”web.gprs.mtnnigeria.net”    //Set GPRS context
        OK
```

In the command above, replace the “*web.gprs.mtnnigeria.net”* with the APN of your SIM network provider (in quotes).

Many network providers require authentication to connect to their APN. You must enter the correct username and password in the following command:

```
AT+XGAUTH=1,1,”web”,”web”
        OK
```

The first string in quotes should be the username and the second is the password.

```
AT+XIIC=1                     //Establish PPP link
        OK
```

```
AT+XIIC?                      //Check state of PPP link
```

```
Possible responses: +XIIC:   1, <IP-address>        OK

                    +XIIC:   0, 0.0.0.0        OK
```

The modem attempts to establish the link with the AT+XIIC=1 command. You must then query it with the second command until it yields a response that contains a valid IP address (like 10.23.233.234). A response containing 0.0.0.0 means it hasn’t connected yet. Keep sending the query periodically for about 15 seconds until it returns an IP address. If it doesn't do so within that interval, re-send the AT+XIIC=1 command and query again. Make sure you followed the steps exactly and that the SIM is registered on the network.

Once you are connected, you can now connect to websites/servers and interact with them. This is done with the TCP/IP protocol. The modem, using the SIM, sets up a TCP link to a given port on a server at a given IP address.

```
AT+TCPSETUP=0,154.183.129.22,6000
        OK
        +TCPSETUP: 0,OK
```

The command above connects to port 6000 of the server at 154.183.129.22 using Link 0 of the modem. The command was successful so the response indicates a TCP link was setup on link 0 successfully. The modem can maintain at most 2 TCP links (link 0 & link 1) at any time. If you want to connect to a website, but don't know its IP address, you can use the 'ping' command in Command Prompt or use a command to the modem that returns the IP address of the site. The following command checks the IP address of Google.com.

```
AT+DNS="www.google.com"
        +DNS: 41.220.75.106
        +DNS: OK
```

The command may return more than one address, depending on the site. Any of them can be used.

Once you have the IP address of the site, to connect to it and receive HTTP pages, you must connect to port 80 of the web server hosting the site using the command given earlier for connecting to servers.

Once you are connected to the server, you can send and receive data from it. To send data to a server you've already connected to, begin like this:

```
AT+TCPSEND=0,12
        > hello, world
        OK
        +TCPSEND:0,12
```

The first line instructs the modem that you want to send 12 bytes of data using link 0. This means that you must already know the length of your data packet in bytes. You must have already established a TCP connection using link 0. When the '>' symbol appears (like when sending an SMS), you can begin typing (in the serial monitor) the data you want to send. Every data packet must end with '\r'. However, the serial monitor's line ending is already set to "Both NL & CR" which automatically attaches '\r\n' to everything you send when you press Enter (or click Send). Therefore, after typing in the monitor, simply press Enter and the data will be sent to the server. You should get the 'OK' response and also confirmation that the right number of bytes have been sent over the link.

This can be used to send HTTP requests to websites in order to, for example, request webpages. Simply connect to the website (port 80) like explained before. Then send the request to the server using the TCPSEND command. One thing to note is that, when using TCPSEND, the modem waits for you to enter up to the number of characters you indicated. E.g. To send a GET request to a server (e.g. store.microscale.net), this is an acceptable format:

```
GET /index.php HTTP/1.1\r\nHost: store.microscale.net\r\n\r\n
```

The request GETs the 'index.php' webpage using HTTP 1.1 protocol. To send this request, you could first send:

```
AT+TCPSEND=0,55          (55 is the length of the GET request)
    > GET /index.php HTTP/1.1
```

When you press Enter, the monitor appends '\r\n' to it, because of the line ending setting. So, the serial monitor helps in completing the request. But because you've not sent up to the 55 bytes promised, the modem still waits, with the '>' symbol, for you to send the rest. Next, you type:

```
    > Host: store.microscale.net
```

You press Enter twice to complete the GET request. Then you press Enter one last time to indicate the end of the data packet to the modem. If it's successful, you should begin receiving data from the web server. Received data (either from a website or from a server you're running) uses the following format:

```
+TCPRECV: <Link>, <data-length>, <Received-data>
```

Data will be received only if the connection is still open. You can check the status of the TCP connection at any time by using this command:

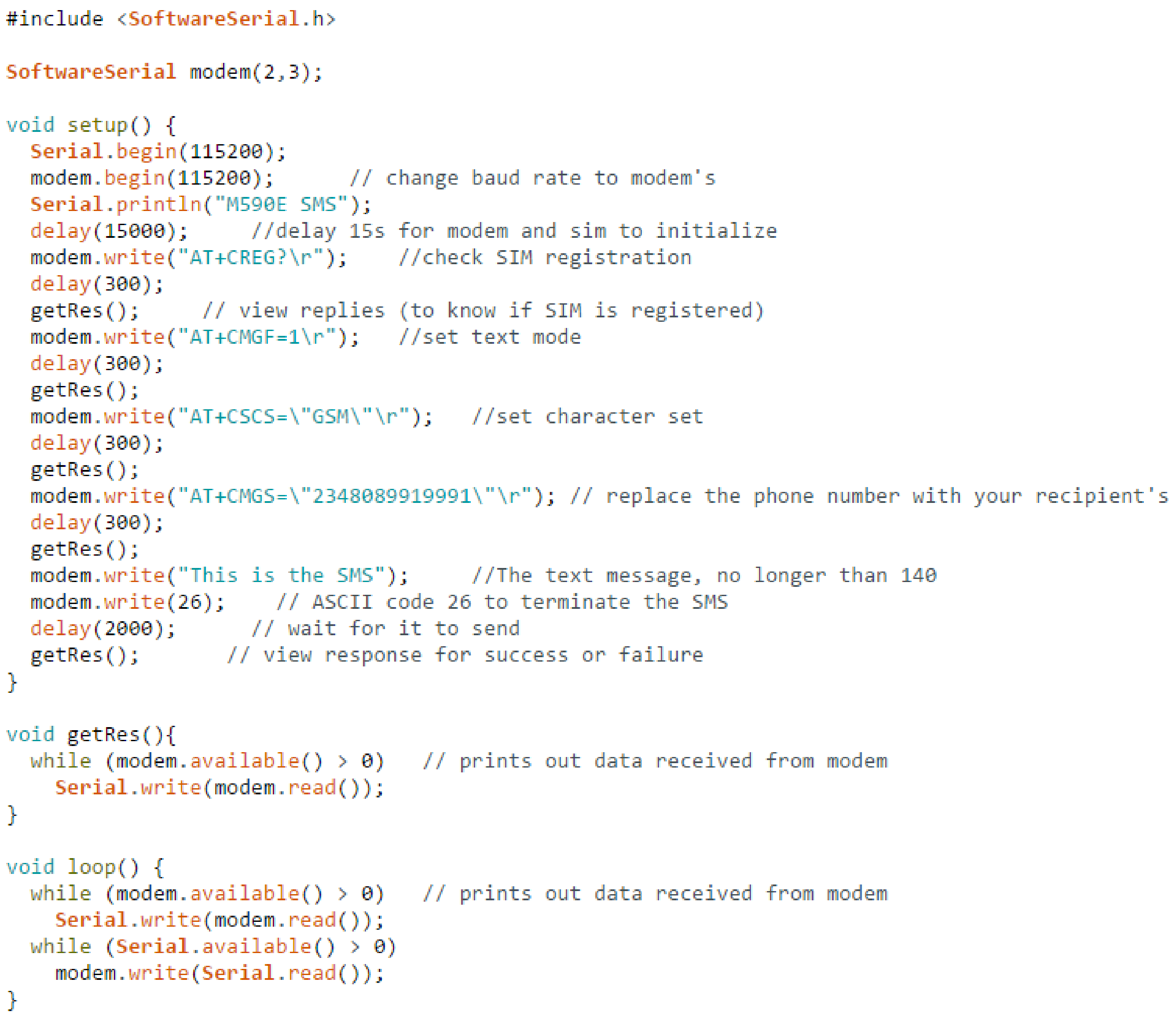
```
AT+IPSTATUS=<Link>
    Possible responses: <Link>, CONNECT, TCP, <port>
                        <Link>, DISCONNECT
```

<Link> stands for the link used to connect to the server (link 0 or 1). <port> stands for the port number you’re connected to on the server. The first response shows you’re still connected while the second shows the modem has been disconnected. A TCP link may be forcibly closed by the server or the modem if it’s unused for a certain amount of time. To close a TCP link when you’re done, use the command:

AT+TCPCLOSE=<Link>

OK

Consult the Neoway documentation for more details on the commands used here and other commands you may be interested in.



```
#include <SoftwareSerial.h>

SoftwareSerial modem(2,3);

void setup() {
  Serial.begin(115200);
  modem.begin(115200);      // change baud rate to modem's
  Serial.println("M590E SMS");
  delay(15000);     //delay 15s for modem and sim to initialize
  modem.write("AT+CREG?\r");    //check SIM registration
  delay(300);
  getRes();     // view replies (to know if SIM is registered)
  modem.write("AT+CMGF=1\r");   //set text mode
  delay(300);
  getRes();
  modem.write("AT+CSCS=\"GSM\"\r");   //set character set
  delay(300);
  getRes();
  modem.write("AT+CMGS=\"2348089919991\"\r"); // replace the phone number with your recipient's
  delay(300);
  getRes();
  modem.write("This is the SMS");     //The text message, no longer than 140
  modem.write(26);    // ASCII code 26 to terminate the SMS
  delay(2000);      // wait for it to send
  getRes();       // view response for success or failure
}

void getRes(){
  while (modem.available() > 0)   // prints out data received from modem
    Serial.write(modem.read());
}

void loop() {
  while (modem.available() > 0)   // prints out data received from modem
    Serial.write(modem.read());
  while (Serial.available() > 0)
    modem.write(Serial.read());
}
```

***Can be used to send SMS. Adjust delays, if need be.

NB: To change the baud rate of the modem (e.g. to 19200), use the command:

```
AT+IPR=19200
        OK
```

To set the modem to auto-baud mode,

```
AT+IPR=0
        OK
```

To save current configuration permanently,

```
AT&W
        OK
```

NOTE:

To perform SMS operations with the Arduino and the modem, you need to increase the receive buffer size of the Software Serial library so that SMS text can be safely received without overflowing the buffer.

Locate the SoftwareSerial.h file in:

C:\Program Files (x86)\Arduino\hardware\arduino\avr\libraries\SoftwareSerial

And change:

```
#define _SS_MAX_RX_BUFF 64 // RX buffer size
```

To:

```
#define _SS_MAX_RX_BUFF 256 // RX buffer size
```

Save the file. This increases the RX buffer size to 256 bytes, enough to safely read SMS and other stuff from the modem, provided you read fast enough from the buffer (overflow is always an issue).

CHALLENGES

- Obtaining a reliable power supply. Finally settled for a 3.7 V Li-Po battery with a 1000uF capacitor helping out.
- Did not follow the power-up and power-down procedure exactly, at first. Caused a lot of unstable behaviour in the modem.

- If you use a bench power supply, never turn it off/on when it’s still connected to the modem. It seems the supply can spike when it’s being turned on/off. First turn on the supply, when it has settled to the desired voltage, connect the modem. To turn off the supply, first disconnect the modem, then turn off the power supply.

CONCLUSION

SMS messages were successfully sent and received with the modem. I also managed to retrieve webpages (like the store.microscale.net/index.php page) and communicate over the internet with a Python server I set up on my PC. Just be careful with the modem and you’ll be fine.

(Brian Ejike)